Multithreaded Web Crawler and Wikipedia Game

Project Overview

In this project, you will develop a **basic multithreaded web crawler** that can fetch multiple web pages simultaneously, parse them for links, and follow those links recursively to find a path between two topics on Wikipedia.

Problem Statement

Traditional single-threaded web crawlers are slow and inefficient for fetching multiple web pages. The objective of this project is to use **multithreading** in **C** to more efficiently crawl a set of URLs.

Wikipedia Game Definition

The Wikipedia Game is a race to find a route between two topics on Wikipedia by traveling through links from article A to article B. For example, a route between the Wikipedia pages for Linux and Rutgers University-Camden is:

```
https://en.wikipedia.org/wiki/Linux
https://en.wikipedia.org/wiki/Bell_Labs
https://en.wikipedia.org/wiki/Murray_Hill,_New_Jersey
https://en.wikipedia.org/wiki/New_Jersey
https://en.wikipedia.org/wiki/Rutgers_University-Camden
```

This path can be followed if you start from the first page and click links to articles found in each article.

Functionality Description

The webcrawler should include the following functionalities:

- **Multithreading:** Fetch multiple web pages concurrently.
- **URL queue**: Implement a thread-safe queue to manage URLs that are pending to be fetched.
- **Error Handling:** Handle errors such as invalid URLs and failed connections.
- **Depth Control:** Allow the crawler to limit the depth of the crawl to prevent infinite recursion.
- **Synchronization:** Implement synchronization mechanisms to manage access to shared resources among threads.

Features Documentation

1. Thread Management
    - Use **pthreads** to handle multiple web page fetches in parallel.
    - Assign each thread a separate URL to process.
    - Make sure threads terminate gracefully once their task is complete.
2. URL Queue
    - Design a thread-safe queue that holds URLs to be crawled.
    - Ensure multiple threads can add to and fetch from the queue.
    - Track depth and parent URL node appropriately, so that the path can be backtracked.
3. HTML Parsing
    - Parse HTML content to extract links.
    - Use **regex**, a different library, or write a custom parser to identify and follow Wikipedia links within the HTML documents.
4. Wikipedia Link Traversal
    - The user should be able to specify two links to Wikipedia articles through command line arguments.
    - Crawl links from the first article, making a maximum amount of link visits equal to the user-specified depth.
    - Output the links the crawler found linking article A to article B if a path is found.
5. Depth Control
    - Implement depth control to limit how deep the crawler goes into a website.
    - The user should be able to specify the maximum depth as a parameter.
6. Synchronization
    - Use mutexes, semaphores, or other synchronization primitives to ensure shared resources like the URL queue are accessed safely.
7. Error handling
    - Implement error handling to manage network failures, invalid URLs, and other exceptions.
    - Ensure the crawler can recover from errors and continue operating.
8. Logging
    - Log the progress of the web crawler, including which URLs have been visited and any errors encountered.

Project Deliverables

You are required to submit a zip file containing the following:

1. **crawler.c** - Source code for the multithreaded web crawler.
2. **Makefile** - A Makefile facilitating compiling and running the web crawler binary.
3. Optionally, any additional C files or modules you create for the project.

Example usage

Help prompt:

```
./crawler -h
```

Output:

```
USAGE: crawler <url-1> <url-2> <depth>
```

Finding a path between the articles for Linux and Rutgers-Camden:

```
./crawler https://en.wikipedia.org/wiki/Linux
↪  https://en.wikipedia.org/wiki/Rutgers_University-Camden 6
```

Output:

```
Finding path from https://en.wikipedia.org/wiki/Linux to
https://en.wikipedia.org/wiki/Rutgers_University-Camden.

https://en.wikipedia.org/wiki/Linux

https://en.wikipedia.org/wiki/Bell_Labs

https://en.wikipedia.org/wiki/Murray_Hill,_New_Jersey

https://en.wikipedia.org/wiki/New_Jersey

https://en.wikipedia.org/wiki/Rutgers_University-Camden
```

Path can't be found within the given depth:

```
./crawler https://en.wikipedia.org/wiki/Linux
↪  https://en.wikipedia.org/wiki/Rutgers_University-Camden 2
```

Output:

```
No path found from https://en.wikipedia.org/wiki/Linux to
↪  https://en.wikipedia.org/wiki/Rutgers_University-Camden.
```

Compilation Requirements

Ensure your project is compilable on **Linux** using the **gcc** compiler.

```
gcc -std=c11 -pedantic -pthread -lcurl crawler.c -o crawler
```

Note:

- `-pthread` is required for multithreading.
- `-lcurl` is required for libcurl networking.

Makefile targets

Your **Makefile** must support:

- `all`: Compiles the web crawler binary.
- `clean`: Removes compiled files.
- `run`: Runs the web crawler with URLs from urls.txt.

Example Makefile:

```
all: gcc -std=c11 -pedantic -pthread -lcurl crawler.c -o crawler
clean: rm -f crawler
run: ./crawler
```